WaveForms SDK User's Manual

Revision: August 2, 2013

DIGILENT®
www.digilentinc.com

1300 NE Henley Court, Suite 3
Pullman, WA 99163
(509) 334 6306 Voice | (509) 334 6300 Fax

Table of Contents

1. Introduction 2
2. Overview 3
2.1 The System 3
2.2 The API 4
2.3 Calling API Functions 5
3. System 6
4. Device Enumeration 7
5. Device Control 9
6. AnalogIn (Oscilloscope) 13
6.1 Control 13
6.2 Configuration 17
6.3 Channels 20
6.4 Trigger 24
7. AnalogOut (Arbitrary Waveform Generator) 34
7.1 Control 35
7.2 Configuration 37
8. AnalogIO 48
9. DigitalIO 53
10. Examples 56
10.1 Device enumeration 56
10.2 Generate sine wave 57
10.3 Generate custom waveform 58
10.4 Single sample read 59
10.5 Acquisition 60
10.6 Analog Discovery Power Supplies 61
10.7 Analog Discovery System Monitor 62
10.8 Digital IO 63

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

1. Introduction

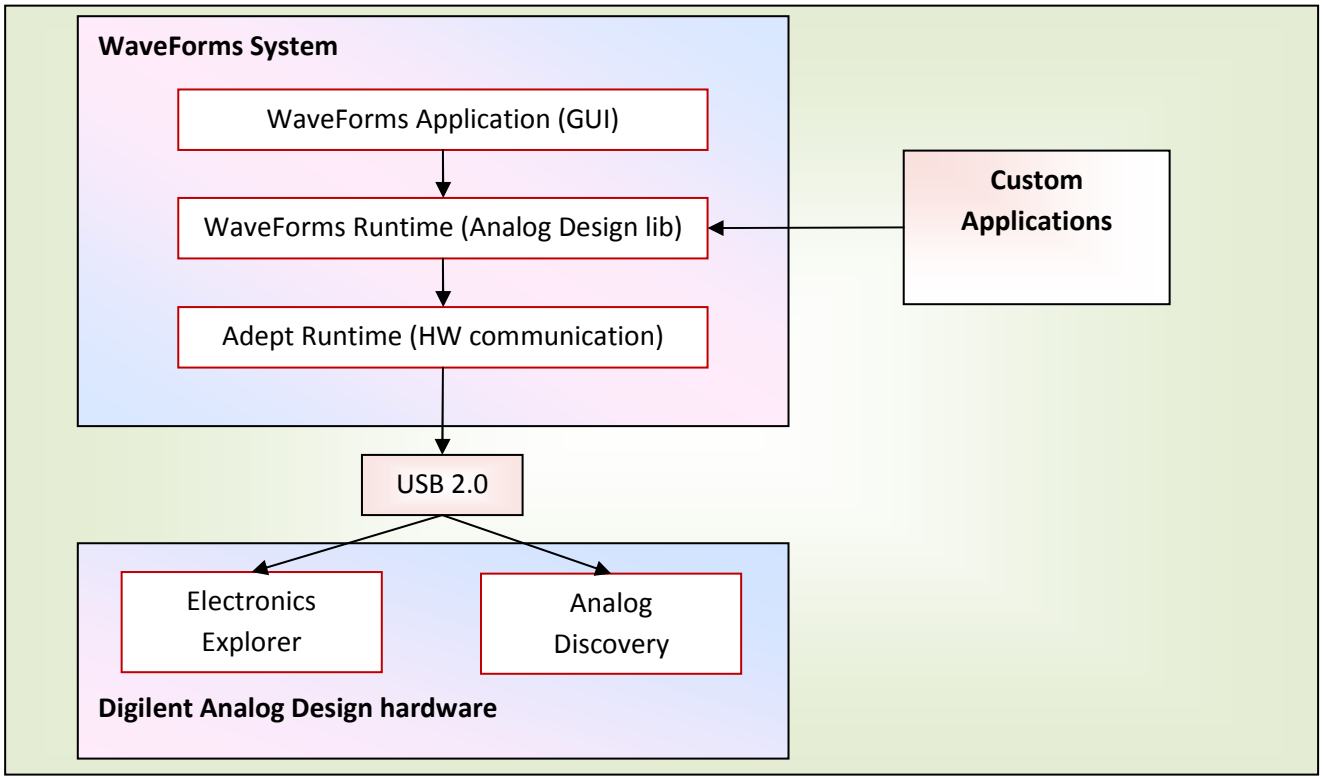
WaveForms™ provides an interface that allows users to interact with Digilent Analog Design hardware such as the Analog Discovery™ and Electronics Explorer™. While the WaveForms application offers a polished graphical interface, the WaveForms SDK exposes a public application programming interface (API) that gives users the ability to create custom PC applications.

This document focuses on the WaveForms SDK. It describes the main components and architecture of the WaveForms system and details each function contained in the WaveForms API. Finally, it offers examples demonstrating how to identify, connect to, and control analog hardware devices.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

2. Overview

This section briefly describes the WaveForms System and provides a basic understanding of the WaveForms API exposed by the SDK.



2.1 The System

As mentioned in the introduction, the WaveForms system is comprised of multiple components. The most visible component is the **WaveForms Application**; a suite of graphical instrument panels that give full access to the analog and digital instruments in the connected hardware. The WaveForms application uses the **WaveForms Runtime** to control all signal generation and acquisition. The WaveForms Runtime is comprised of the *DWF* Dynamic Library and several configuration files. In the Windows version, this is a Dynamic Link Library named **DWF.DLL** and is installed in the System directory (e.g. "C:\Windows\System32"). In the Linux version, this is a shared library named **libdwf.so** and is installed in "/User/Lib".

The DWF Library uses the **Adept Runtime**, which provides basic communication with the targeted hardware instruments (i.e. Analog Discovery and Electronics Explorer). Although the Adept Runtime is an integral part of the WaveForms System, knowledge of its structure *is not required* to write custom applications.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

2.2 The API

Everything needed to write custom applications is included in the **WaveForms SDK,** which provides the header/library files and documentation to access the API for the DWF Library. A custom application must properly link to these files to make the appropriate API function calls. Every function in the WaveForms public API is declared in the **DWF.h** include file.

Basic usage of the WaveForms API can be broken down into the following steps:

1. Call enumeration functions to discover connected hardware devices
2. Call *FDwfDeviceOpen* function to establish a connection to specific hardware device
3. Call function to Enable instrument within hardware device
4. Call functions to configure instrument and acquire/generate signals
5. Call function to Disable instrument
6. Call *FDwfDeviceClose* function to disconnect from device

There are **7** main groups of API functions, each named with different prefixes.

- **Device Enumeration**
  API functions with the prefix *DwfEnum* control the enumeration of Analog Design devices connected to the PC.
- **Device Control**
  API functions with the prefix *DwfDevice* control opening and closing specific Analog Design devices.
- **AnalogIn** (Oscilloscope)
  API functions with a prefix of *DwfAnalogIn* are used to access this instrument.
  This instrument acquires samples from each enabled channel synchronously.
- **AnalogOut** (Arbitrary Waveform Generator)
  API functions with the prefix *DwfAnalogOut* are used to access this instrument.
  This instrument drives signals from each channel independently.
- **AnalogIO**
  API functions with the prefix *DwfAnalogIO* are used to access this instrument.
  This instrument acquires and drives various analog signals.
- **DigitalIO**
  API functions with the prefix *DwfDigitalIO* are used to access this instrument.
  This instrument acquires and drives various digital signals.
- **System**
  API functions with the prefix *DwfGet* are used to obtain basic system information that is instrument and device independent.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

Each instrument is directly controlled using 3 types of functions in the API:

- **Reset function**– This function resets all of the instrument parameters to default values.
  (FDwfAnalogInReset, FDwfAnalogOutReset, FDwfAnalogIOReset, FDwfDigitalIOReset)
- **Configure function** – This function configures and/or starts the instrument.
  (FDwfAnalogInConfigure, FDwfAnalogOutConfigure, FDwfAnalogIOConfigure, FDwfDigitalIOConfigure)
- **Status function** – This function polls and reads all information from the instrument.
  (FDwfAnalogInStatus, FDwfAnalogOutStatus, FDwfAnalogIOStatus, FDwfDigitalIOStatus)
  Note: Although there are multiple "Status" functions for each instrument, these functions are the only ones that actually read data from the device.

There are a number of type definitions and corresponding constants in the DWF.h include file. The majority of them are used as parameters.

When a hardware device is opened, a handle is returned (HDWF) which is used to access and finally close in all instrument API functions.

2.3 Calling API Functions

The API functions are C style and return a Boolean value: TRUE if the call is successful, FALSE if not successful. This Boolean value is an integer type definition, *not* the standard c-type bool. In general, the API functions contain variations of the following parameters:

- ***Info** - Returns detailed information about the parameter support for the instrument (i.e. minimum/maximum values, supported modes, etc.)
- ***Set** - Sets an instrument parameter. When the AutoConfigure is enabled (by default), the instrument is reconfigured and stopped.
- ***Get** - Gets the actual instrument parameter.
- ***Status** - Returns the parameter from the device.

The API functions won't fail when a parameter pointer is NULL or when a setting (*Set) parameter value is out of limits. To verify the actual setting value, se the *Get API return the actual value.

The supported discrete parameters are retrieved in bit field value. To decode the capabilities of the device use the `IsBitSet` macro.

```
int fsfilter;
FDwfAnalogInChannelFilterInfo(h, &fsfilter);
if(IsBitSet(fsfilter, filterAverage)){
    FDwfAnalogInChannelFilterSet(hdwf, 0, filterAverage);
}
```


Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

3. System

```
FDwfGetLastError(DWFERC * pdwferc);
```

Parameters:
- pdwferc - variable to receive error code

This function is used to retrieve the last error code in the calling process. The error code is cleared when other API functions are called and is only set when an API function fails during execution. Error codes are declared in DWF.H:

- dwfercNoErc – No error occurred
- dwfercUnknownError – Call waiting on pending API time out
- dwfercApiLockTimeout – Call waiting on pending API time out
- dwfercAlreadyOpened – Device already opened
- dwfercNotSupported – Device not supported
- dwfercInvalidParameter0 – Parameter 0 was invalid in last API call
- dwfercInvalidParameter1 – Parameter 1 was invalid in last API call
- dwfercInvalidParameter2 – Parameter 2 was invalid in last API call
- dwfercInvalidParameter3 – Parameter 3 was invalid in last API call

```
FDwfGetLastErrorMsg(char szError[512]);
```

Parameters:
- szError - pointer to buffer to receive error string

This function is used to retrieve the last error message. This can consist of a chain of messages, separated by a new line character, that describe the events leading to the failure.

```
FDwfGetVersion(char szVersion[32]);
```

Parameters:
- szVersion - pointer to buffer to receive version string

This function is used to retrieve the version string. The version string is composed of major, minor and build numbers (i.e. "2.0.19").

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

4. Device Enumeration

The *FDwfEnum* functions are used to discover all connected, compatible analog design devices.

```
FDwfEnum(ENUMFILTER enumfilter, int * pnDevice);
```

Parameters:
- enumfilter –filter value to be used for device enumeration. Use the *enumfilterAll* constant to discover all compatible devices.
- pnDevice –integer pointer to return count of found devices by reference

Calling this function will build an internal list of detected devices filtered by the *enumfilter* paramter. This function must be called before using other *FDwfEnum* functions because they obtain information about enumerated devices from this list identified by device index.

```
FDwfEnumDeviceType(int idxDevice, DEVID* pDeviceId, DEVVER* pDeviceRevision);
```

Parameters:
- idxDevice – index of the enumerated device for which to return the type and revision
- pDeviceId – variable to return the device id
- pDeviceRevision – pointer to DEVVER instance to return the device revision by reference

This function is used to return the device ID and version ID.

```
FDwfEnumDeviceIsOpened(int idxDevice, BOOL* pfIsUsed);
```

Parameters:
- idxDevice – index of the enumerated device
- pfIsUsed – pointer to variable to receive Boolean indicating if the device is in use

This function is used to retrieve a Boolean specifying if a device is already opened by this or any other process.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

```
FDwfEnumUserName(int idxDevice, char szUserName[32]);
```

Parameters:
- idxDevice – index of the enumerated device
- szUserName – pointer to character array to return the user name string by reference

This function is used to retrieve the user name of the enumerated device.

```
FDwfEnumDeviceName(int idxDevice, char szDeviceName[32]);
```

Parameters:
- idxDevice – index of the enumerated device
- szDeviceName – pointer to character array to return the device name by reference

This function is used to retrieve the device name of the enumerated device.

```
FDwfEnumSN(int idxDevice, char szSN[32]);
```

Parameters:
- idxDevice – index of the enumerated device
- szSN – pointer to character array to return the serial number by reference

This function is used to retrieve the 12-digit, unique serial number of the enumerated device.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

5. Device Control

```
FDwfDeviceOpen(int idxDevice, HDWF *phdwf);
```

Parameters:
- idxDevice – index of the enumerated device
- phdwf – pointer to *HDWF* variable to receive opened interface handle by reference

This function opens a device identified by the enumeration index and retrieves a handle.
To automatically enumerate all connected devices and open the first discovered device, use index -1.

```
FDwfDeviceClose(HDWF hdwf);
```

Parameters:
- hdwf – interface handle to be closed

This function is used to close an interface handle when access to the device is no longer needed. Once this function has returned, the specified interface handle can no longer be used to access the device.

```
FDwfDeviceCloseAll();
```

Parameters:
None

This function is used to close all opened devices by the calling process. It does not close all devices across all processes.

```
FDwfDeviceAutoConfigureSet(HDWF hdwf, BOOL fAutoConfigure);
```

Parameters:
- hdwf – interface handle
- fAutoConfigure– value for this option: TRUE if enabled, FALSE if disabled.

This function enables or disables the AutoConfig setting for a specific device. When this setting is enabled, the device is automatically configured every time an instrument parameter is set. For example, when AutoConfigure is enabled, FDwfAnalogOutConfigure does *not* need to be called after FDwfAnalogOutRunSet. This adds latency to every Set function; just as much latency as calling the corresponding Configure function directly afterward.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

FDwfDeviceAutoConfigureGet(HDWF hdwf, BOOL* fAutoConfigure);

Parameters:
- hdwf – interface handle
- fAutoConfigure– pointer to variable to receive the current value of this option

This function returns the AutoConfig setting in the device. See the function description for *FDwfDeviceAutoConfigureSet* for details on this setting.

FDwfDeviceReset(HDWF hdwf);

Parameters:
- hdwf – interface handle

This function resets and configures all device and instrument parameters to default values.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

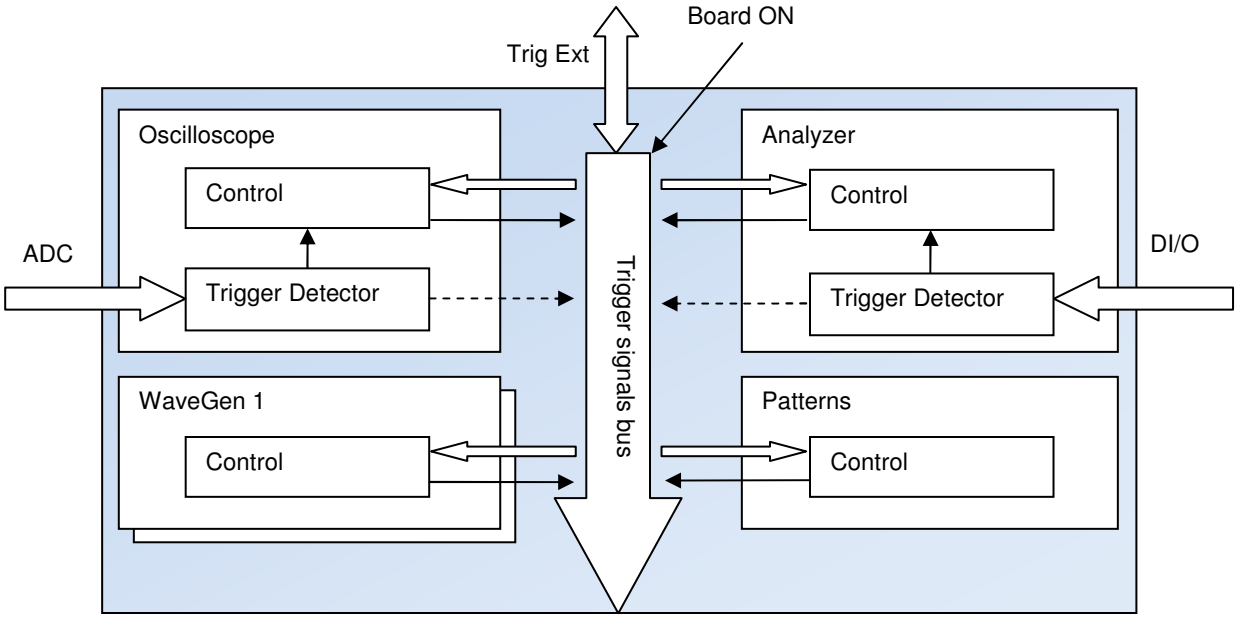
```
FDwfDeviceTriggerInfo(HDWF hdwf, int* pfstrigsrc);
```

Parameters:
- hdwf – interface handle
- pfstrigsrc – variable to receive the supported trigger sources

This function returns the supported trigger source options for the global trigger bus. They are returned (by reference) as a bit field. This bit field can be parsed using the IsBitSet Macro. Individual bits are defined using the TRIGSRC constants in DWH.h.

The global trigger bus allows multiple instruments to trigger each other.
These trigger source options are:

- `trigsrcNone`: (default) the trigger pin is high impedance, input. This is the default setting.
- `trigsrcPC`: trigger from PC, this can be used to synchronously start multiple instruments
- `trigsrcDetectorAnalogIn`: trigger detector on analog input ADC channels
- `trigsrcDetectorDigitalIn`: trigger on digital input channels
- `trigsrcAnalogIn, trigsrcDigitalIn, trigsrcDigitalOut, trigsrcAnalogOut1,2,3,4`: trigger on device instruments, these output high when running
- `trigsrcExternal 1,2,3,4`: external trigger signal



Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

```
FDwfDeviceTriggerSet(HDWF hdwf, int idxPin, TRIGSRC trigsrc);
```

Parameters:
- hdwf – interface handle
- idxPin – external trigger, IO pin index
- trigsrc – trigger source to set

This function is used to configure the global trigger IO pin with a specific TRIGSRC option.

```
FDwfDeviceTriggerGet(HDWF hdwf, int idxPin, TRIGSRC* ptrigsrc);
```

Parameters:
- hdwf – interface handle
- idxPin - external trigger, IO pin index
- ptrigsrc – variable to receive the current trigger source

This function returns the configured trigger setting for a trigger IO pin. The trigger source can be "none" or an internal instrument or external trigger.

```
FDwfDeviceTriggerPC(HDWF hdwf);
```

Parameters:
- hdwf – interface handle

This function generates one pulse on the PC trigger line.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

6. AnalogIn (Oscilloscope)

6.1 Control

```
FDwfAnalogInReset(HDWF hdwf);
```

Parameters:
- hdwf – interface handle

This function resets and configures all AnalogIn instrument parameters to default values.

```
FDwfAnalogInConfigure(HDWF hdwf, BOOL fReconfigure, BOOL fStart);
```

Parameters:
- hdwf – interface handle
- fReconfigure – configure the device
- fStart – start the acquisition

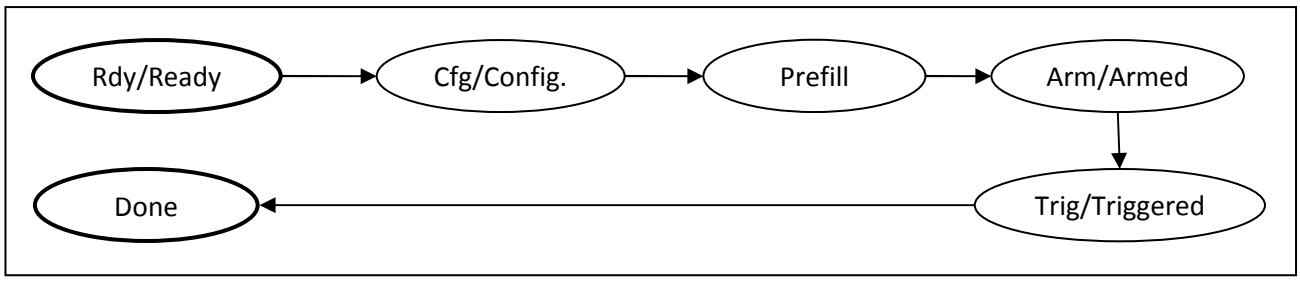
This function is used to configure the instrument and start or stop the acquisition. To reset the Auto trigger timeout, set fReconfigure to TRUE.

```
FDwfAnalogInStatus(HDWF hdwf, BOOL fReadData, STS* psts);
```

Parameters:
- hdwf – interface handle
- fReadData – TRUE if data should be read
- psts – variable to receive the acquisition state

This function is used to check the state of the acquisition. To read the data from the device, set fReadData to TRUE. For single acquisition mode, the data will be read only when the acquisition is finished.

The states of the acquisition:



Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

Note: To ensure simultaneity of information and data, all of the following AnalogInStatus* functions do NOT communicate with the device. These functions only return information and data from the last FDwfAnalogInStatus call.**

States of AnalogIn instrument:

- stsRdy – ready to start
- stsCfg – performing config (offset DA)
- stsPrefill – performs buffer prefill before looking for trigger
- stsArm – armed, waiting for trigger
- stsTrig – triggered
- stsDone – acquisition done

```
FDwfAnalogInStatusSamplesLeft(HDWF hdwf, int* pcSamplesLeft);
```

Parameters:

- hdwf – interface handle
- pcSamplesLeft – variable to receive the remaining samples to acquire

This function is used to retrieve the number of samples left in the acquisition.

```
FDwfAnalogInStatusSamplesValid(HDWF hdwf, int* pcSamplesValid);
```

Parameters:

- hdwf – interface handle
- pcSamplesValid – variable to receive the number of valid samples

This function is used to retrieve the number of valid/acquired data samples.

```
FDwfAnalogInStatusIndexWrite(HDWF hdwf, int* pidxWrite);
```

Parameters:

- hdwf – interface handle
- `pidxWrite` – variable to receive the position of the acquisition

This function is used to retrieve the buffer write pointer. This is needed in ScanScreen acquisition mode to display the scan bar.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

```
FDwfAnalogInStatusAutoTriggered(HDWF hdwf, BOOL* pfAuto);
```

Parameters:
- hdwf – interface handle
- pfAuto – returns TRUE if the acquisition was auto triggered

This function is used to verify if the acquisition is auto triggered.

```
FDwfAnalogInStatusData(HDWF hdwf, int idxChannel, double* rgdVoltData, int
cdData);
```

Parameters:
- hdwf – interface handle
- idxChannel – channel index
- rgdVoltData – pointer to allocated buffer to copy the acquisition data
- cdData – number of samples to copy

This function is used to retrieve the acquired data samples from the specified idxChannel on the AnalogIn instrument. It copies the data samples to the provided buffer.

```
FDwfAnalogInStatusSample(HDWF hdwf, int idxChannel, double* pdVoltSample);
```

Parameters:
- hdwf – interface handle
- idxChannel – channel index
- pdVoltSample – variable to receive the sample value

This function gets the last ADC conversion sample from the specified idxChannel on the AnalogIn instrument.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

```
FDwfAnalogInStatusRecord(HDWF hdwf, int* pcdDataAvailable, int* pcdDataLost, int* pcdDataCorrupt);
```

Parameters:
- hdwf – interface handle
- pcdDataAvailable – pointer to variable to receive the available number of samples
- pcdDataLost – pointer to variable to receive the lost samples after the last check
- pcdDataCorrupt – pointer to variable to receive the number of samples that could be corrupt

This function is used to retrieve information about the recording process. The data lost occurs when the device acquisition is faster than the read process to PC. In this case, the device recording buffer is filled and data samples are overwritten. Corrupt samples indicate that the samples have be overwritten by the acquisition process during the previous read. In this case, try optimizing the loop process for faster execution or reduce the acquisition frequency or record length to be less than or equal to the device buffer size (record length <= buffer size/frequency).

```
FDwfAnalogInRecordLengthSet(HDWF hdwf, double sLegth);
```

Parameters:
- hdwf – interface handle
- sLegth – record length to set expressed in seconds

This function is used to set the Record length in seconds.

```
FDwfAnalogInRecordLengthGet(HDWF hdwf, double* psLegth);
```

Parameters:
- hdwf – interface handle
- sLegth – pointer to variable to receive the record length

This function is used to get the current Record length in seconds.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

6.2 Configuration

```
FDwfAnalogInFrequencyInfo(HDWF hdwf, double* phzMin, double* phzMax);
```

Parameters:
- hdwf – interface handle
- phzMin – pointer to return the minimum allowed frequency
- phzMax – pointer to return the maximum allowed frequency

This function is used to retrieve the minimum and maximum (ADC frequency) settable sample frequency.

```
FDwfAnalogInFrequencySet(HDWF hdwf, double hzFrequency);
```

Parameters:
- hdwf – interface handle
- hzFrequency – acquisition frequency to set

This function is used to set the sample frequency for the AnalogIn instrument.

```
FDwfAnalogInFrequencyGet(HDWF hdwf, double* phzFrequency);
```

Parameters:
- hdwf – interface handle
- phzFrequency – variable to receive the acquisition frequency

This function is used to read the configured sample frequency.
The AnalogIn ADC always runs at maximum frequency, but the method in which the samples are stored in the buffer can be individually configured for each channel with `FDwfAnalogInChannelFilterSet` function.

```
FDwfAnalogInBitsInfo(HDWF hdwf, int* pnBits);
```

Parameters:
- hdwf – interface handle
- pnBits – variable to receive the number of ADC bits

This function is used to retrieve the number bits used by the AnalogIn ADC.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

```
FDwfAnalogInBufferSizeInfo(HDWF hdwf, int* pnSizeMin, int* pnSizeMax);
```

Parameters:
- hdwf – interface handle
- pnMin – pointer to return the minimum buffer size
- pnMax – pointer to return the maximum buffer size

This function returns the minimum and maximum allowable buffer sizes for the AnalogIn instrument

```
FDwfAnalogInBufferSizeSet(HDWF hdwf, int nSize);
```

Parameters:
- hdwf – interface handle
- nSize – buffer size to set

This function is used to adjust the AnalogIn instrument buffer size.

```
FDwfAnalogInBufferSizeGet(HDWF hdwf, int* pnSize);
```

Parameters:
- hdwf – interface handle
- pnSize – variable to receive the current buffer size

This function returns the used AnalogIn instrument buffer size.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

FDwfAnalogInAcquisitionModeInfo(HDWF hdwf, int* pfsacqmode);

Parameters:
- hdwf – interface handle
- pfsacqmode – pointer to return the supported acquisition modes

This function returns the supported AnalogIn acquisition modes. They are returned (by reference) as a bit field. This bit field can be parsed using the IsBitSet Macro. Individual bits are defined using the ACQMODE constants in DWH.h. The acquisition mode selects one of the following modes, ACQMODE:

- acqmodeSingle: perform a single buffer acquisition. This is the default setting.
- acqmodeScanShift: perform a continuous acquisition in FIFO style. The trigger setting is ignored. The last sample is at the end of buffer. The FDwfAnalogInStatusSamplesValid function is used to show the number of the acquired samples, which will grow until reaching the BufferSize. Then the waveform "picture" is shifted for every new sample.
- acqmodeScanScreen: perform continuous acquisition circularly writing samples into the buffer. The trigger setting is ignored. The IndexWrite shows the buffer write position.
  This is similar to a heart monitor display.
- acqmodeRecord: perform acquisition for length of time set by FDwfAnalogInRecordLengthSet.

FDwfAnalogInAcquisitionModeSet(HDWF hdwf, ACQMODE acqmode);

Parameters:
- hdwf – interface handle
- acqmode – acquisition mode to set

This function is used to set the acquisition mode.

FDwfAnalogInAcquisitionModeGet(HDWF hdwf, ACQMODE* pacqmode);

Parameters:
- hdwf – interface handle
- pacqmode – variable to receive the current acquisition mode

This function is used to get retrieve the acquisition mode.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

6.3 Channels

The oscilloscope channel settings are identical across all channels.

```
FDwfAnalogInChannelCount(HDWF hdwf, int* pcChannel);
```

Parameters:
- hdwf – interface handle
- pcChannel – variable to receive the number of channels

This function is used to read the number of AnalogIn channels of the device.

```
FDwfAnalogInChannelEnableSet (HDWF hdwf, int idxChannel, BOOL fEnable);
```

Parameters:
- hdwf – interface handle
- idxChannel – index of channel to enable/disable
- fEnable – set TRUE to enable , FALSE to disable

This function is used to enable or disable the specified AnalogIn channel. The channel must be enabled before acquisition can continue.

```
FDwfAnalogInChannelEnableGet (HDWF hdwf, int idxChannel, BOOL * pfEnable);
```

Parameters:
- hdwf – interface handle
- idxChannel – index of channel
- pfEnable – variable to return enable/disable status of channel

This function is used to get the current enable/disable status of the specified AnalogIn channel.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

FDwfAnalogInChannelFilterInfo(HDWF hdwf, int* pfsfilter);

Parameters:
- hdwf – interface handle
- pfsfilter – pointer to return the supported acquisition modes

This function returns the supported acquisition filters. They are returned (by reference) as a bit field. This bit field can be parsed using the IsBitSet Macro. Individual bits are defined using the FILTER constants in DWH.h. When the acquisition frequency (`FDwfAnalogInFrequencySet`) is less than the ADC frequency (maximum acquisition frequency), the samples can be stored in one of the following ways using FILTER:

- `filterDecimate`: store every Nth ADC conversion, where N = acquisition frequency/ADC frequency
- `filterAverage`: store the average of N ADC conversions
- `filterMinMax`: store interleaved, the minimum and maximum values, of 2xN conversions

FDwfAnalogInChannelFilterSet(HDWF hdwf, int idxChannel, FILTER filter);

Parameters:
- hdwf – interface handle
- idxChannel – channel index
- filter – acquisition sample filter to set

This function is used to set the used acquisition filter for each AnalogIn channel. With channel index -1, every AnalogIn channel filter will be configured to use the same, new option.

FDwfAnalogInChannelFilterGet(HDWF hdwf, int idxChannel, FILTER* pfilter);

Parameters:
- hdwf – interface handle
- idxChannel – channel index
- pfilter– variable to receive the current sample filter

This function returns the configured acquisition filter.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

```
FDwfAnalogInChannelRangeInfo(HDWF hdwf, double* pvoltsMin, double* pvoltsMax,
double* pnSteps);
```

Parameters:
- hdwf – interface handle
- `pvoltsMin` – variable to receive the minimum voltage range
- `pvoltsMax` – variable to receive the maximum voltage range
- pnSteps – variable to receive number of steps

This function returns the minimum and maximum range, peak to peak values, and the number of adjustable steps.

```
FDwfAnalogInChannelRangeSteps(HDWF hdwf, double rgVoltsStep[32], int* pnSteps);
```

Parameters:
- hdwf – interface handle
- rgVoltsStep – pointer to buffer to receive the range steps
- pnSteps – variable to receive number range steps

This function is used to read the range of steps supported by the device. For instance: 1, 2, 5, 10, etc.

```
FDwfAnalogInChannelRangeSet(HDWF hdwf, int idxChannel, double voltsRange);
```

Parameters:
- hdwf – interface handle
- idxChannel – channel index
- voltsRange – voltage range to set

This function is used to configure the range for each channel.
With channel index -1 every AnalogIN channel range will be configured to the same, new value.

```
FDwfAnalogInChannelRangeGet(HDWF hdwf, int idxChannel, double* pvoltsRange);
```

Parameters:
- hdwf – interface handle
- idxChannel – channel index
- pvoltsRange – variable to receive the current voltage range

This function returns the configured range value for the given channel index

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

```
FDwfAnalogInChannelOffsetInfo(HDWF hdwf, double* pvoltsMin, double* pvoltsMax,
double* pnSteps);
```

Parameters:

- hdwf – interface handle
- `pvoltsMin` – variable to receive the minimum offset voltage
- `pvoltsMax` – variable to receive the maximum offset voltage
- pnSteps – variable to receive the number offset steps

This function returns the minimum and maximum offset levels supported, and the number of adjustable steps.

```
FDwfAnalogInChannelOffsetSet(HDWF hdwf, int idxChannel, double voltOffset);
```

Parameters:

- hdwf – interface handle
- idxChannel – channel index
- voltsRange – channel offset voltage to set

This function is used to configure the offset for each channel. When channel index is specified as -1, every AnalogIn channel offset will be configured to the same level.

```
FDwfAnalogInChannelOffsetGet(HDWF hdwf, int idxChannel, double* pvoltOffset);
```

Parameters:

- hdwf – interface handle
- idxChannel – channel index
- pvoltsRange – variable to receive the offset voltage obtained

This function returns for each AnalogIn channel the configured offset level.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

6.4 Trigger

The trigger is used for Single and Record acquisitions. For ScanScreen and ScanShift, the trigger is ignored.

```
FDwfAnalogInTriggerSourceInfo(HDWF hdwf, int* pfstrigsrc);
```

Parameters:
- hdwf – interface handle
- pfstrigsrc – variable to receive the supported trigger sources

This function returns the supported trigger source options for the AnalogIn instrument. They are returned (by reference) as a bit field. This bit field can be parsed using the IsBitSet Macro. Individual bits are defined using the TRIGSRC constants in DWH.h. For descriptions of these constants, see the description of **FDwfDeviceTriggerInfo**.

```
FDwfAnalogInTriggerSourceSet(HDWF hdwf, TRIGSRC trigsrc);
```

Parameters:
- hdwf – interface handle
- trigsrc – trigger source to set

This function is used to configure the AnalogIn acquisition trigger source.

```
FDwfAnalogInTriggerSourceGet(HDWF hdwf, TRIGSRC* ptrigsrc);
```

Parameters:
- hdwf – interface handle
- ptrigsrc – variable to receive the current trigger source

This function returns the configured trigger source. The trigger source can be "none" or an internal instrument or external trigger. To use the trigger on AnalogIn channels (edge, pulse, etc.), use trigsrcDetectorAnalogIn.

```
FDwfAnalogInTriggerPositionInfo(HDWF hdwf, double* psecMin, double* psecMax);
```

Parameters:
- hdwf – interface handle
- psecMin – variable to receive the minimum trigger position
- psecMax – variable to receive the maximum trigger position

This function returns the minimum and maximum values of the trigger position in seconds.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

```
FDwfAnalogInTriggerPositionSet(HDWF hdwf, double secPosition);
```

Parameters:
- hdwf – interface handle
- secPosition – trigger position to set

This function is used to configure the horizontal trigger position in seconds.

```
FDwfAnalogInTriggerPositionGet(HDWF hdwf, double* psecPosition);
```

Parameters:
- hdwf – interface handle
- psecPosition – variable to receive the current trigger position

This function returns the configured trigger position in seconds.
The horizontal trigger position is used for Single acquisition mode and is relative to the buffer middle point.

```
FDwfAnalogInTriggerAutoTimeoutInfo(HDWF hdwf, double* psecMin, double* psecMax,
int* pnSteps);
```

Parameters:
- hdwf – interface handle
- psecMin – variable to receive the minimum timeout
- psecMax – variable to receive the maximum timeout
- pnSteps – variable to return the number of steps

This function returns the minimum and maximum auto trigger timeout values, and the number of adjustable steps.

```
FDwfAnalogInTriggerAutoTimeoutSet(HDWF hdwf, double secTimeout);
```

Parameters:
- hdwf – interface handle
- secTimeout – timeout to set

This function is used to configure the auto trigger timeout value in seconds.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

```
FDwfAnalogInTriggerAutoTimeoutGet(HDWF hdwf, double* psecTimeout);
```

Parameters:
- hdwf – interface handle
- psecTimeout – variable to receive the current timeout

This function returns the configured auto trigger timeout value in seconds.
The acquisition is auto triggered when the specified time elapses. With zero value the timeout is disabled, performing "Normal" acquisitions.

```
FDwfAnalogInTriggerHoldOffInfo(HDWF hdwf, double* psecMin, double* psecMax);
```

Parameters:
- hdwf – interface handle
- psecMin – variable to receive the minimum holdoff value
- psecMax – variable to receive the maximum holdoff value

This function returns the supported range of the trigger Hold-Off time in Seconds. The trigger hold-off is an adjustable period of time during which the acquisition will not trigger. This feature is used when you are triggering on burst waveform shapes, so the oscilloscope triggers only on the first eligible trigger point.

```
FDwfAnalogInTriggerHoldOffSet(HDWF hdwf, double secHoldOff);
```

Parameters:
- hdwf – interface handle
- secHoldOff – holdoff to set

This function is used to set the trigger hold-off for the AnalongIn instrument in Seconds.

```
FDwfAnalogInTriggerHoldOffGet(HDWF hdwf, double* psecHoldOff);
```

Parameters:
- hdwf – interface handle
- psecHoldOff – variable to receive the current holdoff value

This function is used to get the current trigger hold-off for the AnalongIn instrument in Seconds.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

FDwfAnalogInTriggerTypeInfo(HDWF hdwf, int* pfstrigtype);

Parameters:
- hdwf – interface handle
- pfstrigtype – variable to receive the supported trigger types

This function returns the supported trigger type options for the AnalogIn instrument. They are returned (by reference) as a bit field. This bit field can be parsed using the IsBitSet Macro. Individual bits are defined using the TRIGTYPE constants in DWH.h. These trigger type options are:
- trigtypeEdge: trigger on rising or falling edge. This is the default setting.
- trigtypePulse: trigger on positive or negative; less, timeout or more pulse lengths
- trigtypeTransition: trigger on rising or falling; less, timeout or more transition times

FDwfAnalogInTriggerTypeSet(HDWF hdwf, TRIGTYPE trigtype);

Parameters:
- hdwf – interface handle
- trigtype – trigger type to set

This function is used to set the trigger type for the AnalongIn instrument.

FDwfAnalogInTriggerTypeGet(HDWF hdwf, TRIGTYPE* ptrigtype);

Parameters:
- hdwf – interface handle
- ptrigtype – variable to receive the current trigger type

This function is used to get the current trigger type for the AnalongIn instrument.

FDwfAnalogInTriggerChannelInfo(HDWF hdwf, int* pidxMin, int* pidxMax);

Parameters:
- hdwf – interface handle
- pidxMin – variable to receive the minimum channel index
- pidxMax – variable to receive the maximum channel index

This function returns the range of channels that can be triggered on.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

```
FDwfAnalogInTriggerChannelSet(HDWF hdwf, int idxChannel);
```

Parameters:
- hdwf – interface handle
- idxChannel – trigger channel index to set

This function is used to set the trigger channel.

```
FDwfAnalogInTriggerChannelGet(HDWF hdwf, int* pidxChannel);
```

Parameters:
- hdwf – interface handle
- pidxChannel – variable to receive the current trigger channel index

This function is used to retrieve the current trigger channel index.

```
FDwfAnalogInTriggerFilterInfo(HDWF hdwf, int* pfsfilter);
```

Parameters:
- hdwf – interface handle
- pfsFilter – variable to receive the supported trigger filters

This function returns the supported trigger filters. They are returned (by reference) as a bit field. This bit field can be parsed using the IsBitSet Macro. Individual bits are defined using the FILTER constants in DWH.h.
Select trigger detector sample source, FILTER:
- filterDecimate: looks for trigger in each ADC conversion, can detect glitches
- filterAverage: looks for trigger only in average of N samples, given by FDwfAnalogInFrequencySet

```
FDwfAnalogInTriggerFilterSet(HDWF hdwf, FILTER filter);
```

Parameters:
- hdwf – interface handle
- filter – trigger filter to set

This function is used to set the trigger filter.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

FDwfAnalogInTriggerFilterGet(HDWF hdwf, FILTER* pfilter);

Parameters:
- hdwf – interface handle
- pfilter – variable to receive the current trigger filter

This function is used to get the trigger filter.

FDwfAnalogInTriggerLevelInfo(HDWF hdwf, double* pvoltsMin, double* pvoltsMax, int* pnSteps);

Parameters:
- hdwf – interface handle
- pvoltsMin – variable to receive the minimum voltage level
- pvoltsMax – variable to receive the maximum voltage level
- pnSteps – variable to receive the number of voltage level steps

This function is used to retrieve the range of valid trigger voltage levels for the AnalogIn instrument in Volts.

FDwfAnalogInTriggerLevelSet(HDWF hdwf, double voltsLevel);

Parameters:
- hdwf – interface handle
- voltsLevel – trigger voltage level to set

This function is used to set the trigger voltage level in Volts.

FDwfAnalogInTriggerLevelGet(HDWF hdwf, double* pvoltsLevel);

Parameters:
- hdwf – interface handle
- pvoltsLevel – variable to receive the current trigger voltage level

This function is used to get the current trigger voltage level in Volts.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

FDwfAnalogInTriggerHysteresisInfo(HDWF hdwf, double* pvoltsMin, double* pvoltsMax, int* pnSteps);

Parameters:
- hdwf – interface handle
- pvoltsMin – variable to receive the minimum hysteresis level
- pvoltsMax – variable to receive the maximum hysteresis level
- pnSteps – variable to receive the number of hysteresis level steps

This function is used to retrieve the range of valid trigger hysteresis voltage levels for the AnalogIn instrument in Volts. The trigger detector uses two levels: low level (TriggerLevel - Hysteresis) and high level (TriggerLevel + Hysteresis). Trigger hysteresis can be used to filter noise for Edge or Pulse trigger. The low and high levels are used in transition time triggering.

FDwfAnalogInTriggerHysteresisSet(HDWF hdwf, double voltsLevel);

Parameters:
- hdwf – interface handle
- voltsLevel – trigger hysteresis level to set

This function is used to set the trigger hysteresis level in Volts.

FDwfAnalogInTriggerHysteresisGet(HDWF hdwf, double* pvoltsHysteresis);

Parameters:
- hdwf – interface handle
- pvoltsLevel – variable to receive the current trigger hysteresis level

This function is used to get the current trigger hysteresis level in Volts.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

```
FDwfAnalogInTriggerConditionInfo(HDWF hdwf, int* pfstrigcond);
```

Parameters:
- hdwf – interface handle
- pfstrigcond – variable to receive the supported trigger conditions

This function returns the supported trigger type options for the AnalogIn instrument. They are returned (by reference) as a bit field. This bit field can be parsed using the IsBitSet Macro. Individual bits are defined using the TRIGCOND constants in DWH.h. These trigger condition options are:

trigcondRisingPositive (This is the default setting):
- for edge and transition trigger on rising edge
- for pulse trigger on positive pulse

- trigcondFallingNegative:
  - for edge and transition trigger on falling edge
  - for pulse trigger on negative pulse

```
FDwfAnalogInTriggerConditionSet(HDWF hdwf, TRIGCOND trigcond);
```

Parameters:
- hdwf – interface handle
- trigcond – trigger condition to set

This function is used to set the trigger condition for the AnalongIn instrument.

```
FDwfAnalogInTriggerConditionGet(HDWF hdwf, TRIGCOND* ptrigcond);
```

Parameters:
- hdwf – interface handle
- ptrigcond – variable to receive the current trigger condition

This function is used to set the trigger condition for the AnalongIn instrument.

```
FDwfAnalogInTriggerLengthInfo(HDWF hdwf, double* psecMin, double* psecMax);
```

Parameters:
- hdwf – interface handle
- psecMin – variable to receive the minimum trigger length
- psecMax – variable to receive the maximum trigger length

This function returns the supported trigger range of trigger length for the AnalogIn instrument in Seconds.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

FDwfAnalogInTriggerLengthSet(HDWF hdwf, double secLength);

Parameters:
- hdwf – interface handle
- secLength – trigger length to set

This function is used to set the trigger length for the AnalongIn instrument in Seconds.

FDwfAnalogInTriggerLengthGet(HDWF hdwf, double* psecLength);

Parameters:
- hdwf – interface handle
- secLength – variable to receive the current trigger length

This function is used to get the current trigger length for the AnalongIn instrument in Seconds.

FDwfAnalogInTriggerLengthConditionInfo(HDWF hdwf, int* pfstriglen);

Parameters:
- hdwf – interface handle
- pfsstriglen – variable to receive the supported trigger length conditions

This function returns the supported trigger length condition options for the AnalogIn instrument. They are returned (by reference) as a bit field. This bit field can be parsed using the IsBitSet Macro. Individual bits are defined using the TRIGLEN constants in DWH.h. These trigger length condition options are:
- triglenLess: trigger immediately when a shorter pulse or transition time is detected
- triglenTimeout: trigger immediately as the pulse length or transition time is reached
- triglenMore: trigger when the length/time is reached and pulse or transition has ended

FDwfAnalogInTriggerLengthConditionSet(HDWF hdwf, TRIGLEN triglen);

Parameters:
- hdwf – interface handle
- triglen – trigger length condition to set

This function is used to set the trigger length condition for the AnalongIn instrument.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

FDwfAnalogInTriggerLengthConditionGet(HDWF hdwf, TRIGLEN* ptriglen);

Parameters:

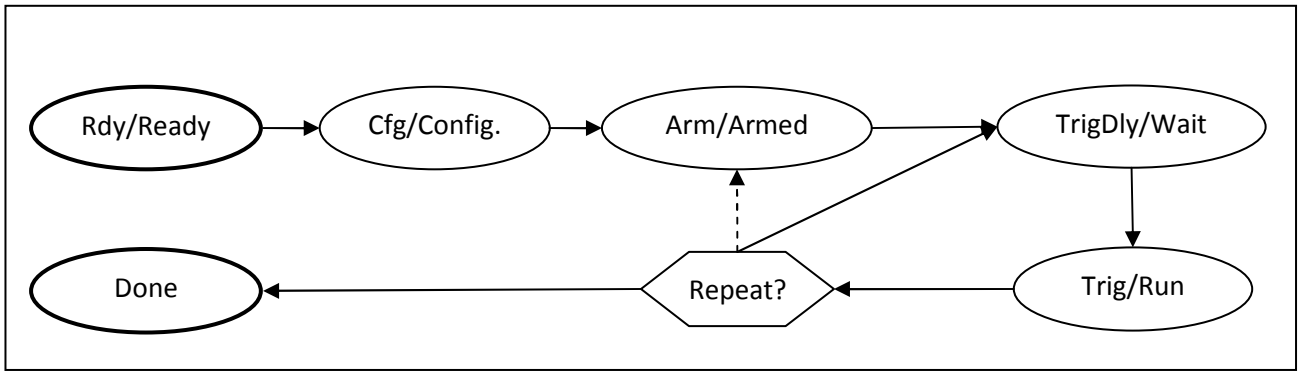
- hdwf – interface handle
- ptriglen – variable to receive the current trigger length condition

This function is used to get the current trigger length condition for the AnalongIn instrument.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

7. AnalogOut (Arbitrary Waveform Generator)

The analog out channels can run independently or synchronized.



States of AnalogOut instrument:

- stsRdy – ready to start
- stsCfg – performing configuration (offset DA…)
- stsArm – armed, waiting for trigger
- stsTrigDly – wait the specified time before running
- stsTrig – run the specified amount of time
    - Repeat – repeat the wait-run or trig-wait-run sequence for the number of specified time
- stsDone – done

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

7.1 Control

FDwfAnalogOutReset(HDWF hdwf, int idxChannel);
Parameters:
- hdwf – interface handle
- idxChannel – channel index

This function resets and configures all AnalogOut instrument parameters to default values for the specified channel. To reset instrument parameters across all channels, set idxChannel to -1.

FDwfAnalogOutConfigure(HDWF hdwf, int idxChannel, BOOL fStart);

Parameters:
- hdwf – interface handle
- idxChannel – channel index
- fStart – start the acquisition. To stop, set to FALSE.

This function is used to configure the instrument and start or stop the AnalogOut instrument. If idxChannel is set to -1, the enabled AnalogOut channels will run synchronized on the same trigger, wait, run and then repeat the configuration

FDwfAnalogOutStatus(HDWF hdwf, int idxChannel, STS* psts);

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- psts – pointer to variable to return the state

This function is used to check the state of the AnalogOut instrument. Channels index -1 can be used when the channels were started synchronously and have the same frequency setting.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

```
FDwfAnalogOutPlayStatus(HDWF hdwf, int idxChannel, int* cdDataFree, int
*cdDataLost, int *cdDataCorrupted);
```

Parameters:

- hdwf – open interface handle on a device
- idxChannel – channel index
- cdDataFree – pointer to variable to return the available free buffer space, the number of new samples that can be sent
- cdDataLost – pointer to variable to return the number of lost samples
- cdDataCorrupted – pointer to variable to return the number of samples that could be corrupted

This function is used to retrieve information about the play process.
Channel index -1 can be used when these were started synchronously and have the same frequency setting. The data lost occurs when the device generator is faster than the sample send process from the PC. In this case, the device buffer gets emptied and generated samples are repeated. Corrupt samples are a warning that the buffer might have been emptied while samples were sent to the device. In this case, try optimizing the loop for faster execution; or reduce the frequency or run time to be less or equal to the device buffer size (run time <= buffer size/frequency).

```
FDwfAnalogOutPlayData(HDWF hdwf, int idxChannel, double* rgdData, int cdData);
```

Parameters:

- hdwf – open interface handle on a device
- idxChannel – channel index
- rgdData – pointer to samples array to be send to the device
- cdData – number of samples to send

This function is used to sending new data samples for play mode. Before starting the Analog Out instrument, prefill the device buffer with the first set of samples using the AnalogOutDataSet function. In the loop of sending the following samples, first call AnalogOutStatus to read the information from the device, then AnalogOutPlayStatus to find out how many new samples can be sent, then send the samples with AnalogOutPlayData.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

7.2 Configuration

```
FDwfAnalogOutCount(HDWF hdwf, int* pcChannel);
```

Parameters:
- hdwf – open interface handle on a device
- pcChannel – pointer to variable to receive the number of channels in the instrument

This function returns the number of Analog Out channels by the device specified by hdwf.

```
FDwfAnalogOutEnableSet(HDWF hdwf, int idxChannel, BOOL fEnable);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- fEnable – TRUE to enable, FALSE to disable

This function enables or disables the channel specified by idxChannel.

```
FDwfAnalogOutEnableGet(HDWF hdwf, int idxChannel, BOOL* pfEnable);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- pfEnable – pointer to variable to receive enabled state

This function is used to verify if a specific channel is enabled or disabled.

```
FDwfAnalogOutTriggerSourceInfo(HDWF hdwf, int idxChannel, int* pfstrigsrc);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- pfstrigsrc – variable to receive the supported trigger sources

This function returns the supported trigger source options for the AnalogOut instrument. They are returned (by reference) as a bit field. This bit field can be parsed using the IsBitSet Macro. Individual bits are defined using the TRIGSRC constants in DWH.h. See the description of **FDwfDeviceTriggerInfo**.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

```
FDwfAnalogOutTriggerSourceSet(HDWF hdwf, int idxChannel, TRIGSRC trigsrc);
```
Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- trigsrc – trigger source to set

This function is used to set the trigger source for the AnalogOut instrument.

```
FDwfAnalogOutTriggerSourceGet(HDWF hdwf, int idxChannel, TRIGSRC* ptrigsrc);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- ptrigsrc – pointer to variable to receive the trigger source

This function is used to get the current trigger source setting for the AnalogOut instrument.

```
FDwfAnalogOutRunInfo(HDWF hdwf, int idxChannel, double* psecMin, double* psecMax);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- psecMin – variable to receive the supported minimum run length
- psecMax – variable to receive the supported maximum run length

This function is used to return the supported run length range for the AnalogOut instrument in Seconds.

```
FDwfAnalogOutRunSet(HDWF hdwf, int idxChannel, double secRun);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- secRun – run length to set expressed in seconds

This function is used to set the run length for the AnalogOut instrument in Seconds. Zero values represent an infinite (or continuous) run.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

```
FDwfAnalogOutRunGet(HDWF hdwf, int idxChannel, double* psecRun);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- psecRun – pointer to variable to receive the run length

This function is used to read the configured run length for the AnalogOut instrument in Seconds.

```
FDwfAnalogOutRunStatus(HDWF hdwf, int idxChannel, double* psecRun);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- psecRun – pointer to variable to receive the remaining run length

This function is used to read the remaining run length. It returns data from the last FDwfAnalogOutStatus call.

```
FDwfAnalogOutWaitInfo(HDWF hdwf, int idxChannel, double* psecMin, double*
psecMax);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- psecMin – variable to receive the supported minimum wait length
- psecMax – variable to receive the supported maximum wait length

This function is used to return the supported wait length range in Seconds. The weight length is how long the AnalogOut instrument waits after being triggered to generate the signal.

```
FDwfAnalogOutWaitSet(HDWF hdwf, int idxChannel, double secWait);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- secWait – wait length to set expressed in seconds

This function is used to set the wait length. Zero value represents no wait.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

```
FDwfAnalogOutWaitGet(HDWF hdwf, int idxChannel, double* psecWait);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- psecWait – pointer to variable to receive the wait length

This function is used to get the current wait length for the AnalogOut instrument.

```
FDwfAnalogOutRepeatInfo(HDWF hdwf, int idxChannel, int* pnMin, int* pnMax);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- pnMin – variable to receive the supported minimum repeat count
- pnMax – variable to receive the supported maximum repeat count

This function is used to return the supported repeat count range. This is how many times the generated signal will be repeated upon play.

```
FDwfAnalogOutRepeatSet(HDWF hdwf, int idxChannel, int cRepeat);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- cRepeat – repeat count to set

This function is used to set the repeat count. Zero value represents infinite repeat.

```
FDwfAnalogOutRepeatGet(HDWF hdwf, int idxChannel, int* pcRepeat);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- pcRepeat – pointer to variable to receive the repeat count

This function is used to read the current repeat count.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

```
FDwfAnalogOutRepeatStatus(HDWF hdwf, int idxChannel, int* pcRepeat);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- pcRepeat – pointer to variable to receive the remaining repeat counts

This function is used to read the remaining repeat counts. It only returns information from the last FDwfAnalogOutStatus function call, it does not read from the device.

```
FDwfAnalogOutRepeatTriggerSet(HDWF hdwf, int idxChannel, BOOL fRepeatTrigger);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- fRepeatTrigger – Boolean used to specify if the trigger should be included in a repeat cycle

This function is used to set the repeat trigger option. To include the trigger in wait-run repeat cycles, set fRepeatTrigger to TRUE. It is disabled by default.

```
FDwfAnalogOutRepeatTriggerGet(HDWF hdwf, int idxChannel, BOOL* pfRepeatTrigger);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- pfRepeatTrigger – pointer to variable to receive the repeat trigger option

This function is used to verify if the trigger has been included in wait-run repeat cycles.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

```
FDwfAnalogOutFunctionInfo(HDWF hdwf, int idxChannel, int* pfsfunc);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- pfsfunc – variable to receive the supported generator function options

This function returns the supported AnalogOut generator function options. They are returned (by reference) as a bit field. This bit field can be parsed using the IsBitSet Macro. Individual bits are defined using the FUNC constants in DWH.h. These are:

- funcDC: generate DC value set as offset
- funcSine: generate sine waveform
- funcSquare: generate square waveform
- funcTriangle: generate triangle waveform
- funcRampUp: generate a waveform with a ramp-up voltage at the beginning
- funcRampDown: generate a waveform with a ramp-down voltage at the end
- funcNoise: generate noise waveform from random samples
- funcCustom: generate waveform from custom repeated data
- funcPlay: generate waveform from custom data in stream play style

```
FDwfAnalogOutFunctionSet(HDWF hdwf, int idxChannel, FUNC func);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- func – generator function option to set

This function is used to set the generator output function for the specified AnalogOut instrument channel.

```
FDwfAnalogOutFunctionGet(HDWF hdwf, int idxChannel, FUNC* pfunc);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- ptrigsrc – pointer to variable to receive the generator function option

This function is used to retrieve the current generator function option for the specified AnalogOut instrument channel.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

```
FDwfAnalogOutFrequencyInfo(HDWF hdwf, int idxChannel, double* phzMin, double* phzMax);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- phzMin – variable to receive the supported minimum frequency
- phzMax – variable to receive the supported maximum frequency

This function is used to return the supported frequency range for the AnalogOut Instrument. The maximum value shows the DAC frequency.

```
FDwfAnalogOutFrequencySet(HDWF hdwf, int idxChannel, double hzFrequency);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- hzFrequency – frequency value to set expressed in Hz

This function is used to set the frequency.

```
FDwfAnalogOutFrequencyGet(HDWF hdwf, int idxChannel, double* phzFrequency);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- hzFrequency – pointer to variable to receive frequency value in Hz

The frequency of the generated waveform: repetition frequency for standard types and custom data; DAC update for noise type; sample rate for play type.

```
FDwfAnalogOutAmplitudeInfo(HDWF hdwf, int idxChannel, double* pvoltsMin, double* pvoltsMax);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- pvoltsMin – minimum amplitude level
- pvoltsMax – maximal amplitude level

This function is used to retrieve the amplitude range for the specified channel on the AnalogOut instrument.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

```
FDwfAnalogOutAmplitudeSet(HDWF hdwf, int idxChannel, double voltsAmplitude);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- voltsAmplitude – amplitude of channel in Volts
-

This function is used to set the amplitude for the specified channel on the AnalogOut instrument in Volts.

```
FDwfAnalogOutAmplitudeGet(HDWF hdwf, int idxChannel, double* pvoltsAmplitude);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- pvoltsAmplitude – pointer to variable to receive amplitude value in Volts
-

This function is used to get the currently set amplitude for the specified channel on the AnalogOut instrument in Volts.

```
FDwfAnalogOutOffsetInfo(HDWF hdwf, int idxChannel, double* pvoltsMin, double*
pvoltsMax);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- pvoltsMin – minimum offset voltage
- pvoltsMax – maximum offset voltage

This function is used to retrieve the offset voltage range for the specified channel on the AnalogOut instrument.

```
FDwfAnalogOutOffsetSet(HDWF hdwf, int idxChannel, double voltsOffset);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- voltsOffset – value to set voltage offset in Volts
-

This function is used to set the offset voltage value for the specified channel on the AnalogOut instrument.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

```
FDwfAnalogOutOffsetGet(HDWF hdwf, int idxChannel, double* pvoltsOffset);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- pvoltsOffset – pointer to variable to receive offset value in Volts

This function is used to get the current offset voltage value for the specified channel on the AnalogOut instrument.

```
FDwfAnalogOutSymmetryInfo(HDWF hdwf, int idxChannel, double* ppercentageMin,
double* ppercentageMax);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- ppercentageMin – minimum value of Symmetry percentage
- ppercentageMax – maximum value of Symmetry percentage

This function is used to obtain the symmetry (or duty cycle) range of the AnalogOut instrument. This symmetry is the pulse duration divided by the pulse period.

```
FDwfAnalogOutSymmetrySet(HDWF hdwf, int idxChannel, double percentageSymmetry);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- percentageSymmetry –value of percentage of Symmetry (duty cycle)

This function is used to set the symmetry (or duty cycle) for the specified channel on the AnalogOut instrument.

```
FDwfAnalogOutSymmetryGet(HDWF hdwf, int idxChannel, double* ppercentageSymmetry);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- ppercentageSymmetry –– pointer to variable to receive value of Symmetry (duty cycle)

This function is used to get the currently set symmetry (or duty cycle) for the specified channel of the AnalogOut instrument.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

FDwfAnalogOutPhaseInfo(HDWF hdwf, int idxChannel, double* pdegreeMin, double* pdegreeMax);

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- pdegreeMin – minimum value of Phase (in degrees)
- pdegreeMax – maximum value of Phase (in degrees)

This function is used to retrieve the Phase range (in degrees) for the specified channel of the AnalogOut instrument.

FDwfAnalogOutPhaseSet(HDWF hdwf, int idxChannel, double degreePhase);

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- degreePhase – value of Phase in degrees

This function is used to set the phase for the specified channel on the AnalogOut instrument.

FDwfAnalogOutPhaseGet(HDWF hdwf, int idxChannel, double* pdegreePhase);

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- pdegreePhase – pointer to variable to receive Phase value (in degrees)

This function is used to get the current phase for the specified channel on the AnalogOut instrument.

FDwfAnalogOutDataInfo(HDWF hdwf, int idxChannel, int* pnSamplesMin, double* pnSamplesMax);

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- pnSamplesMin - minimum number of samples available for custom data
- pnSamplesMax – maximum number of samples available for custom data

This function is used to retrieve the minimum and maximum number of samples allowed for custom data generation.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

```
FDwfAnalogOutDataSet(HDWF hdwf, int idxChannel, double* rgdData, int cdData);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- rgbData – buffer of samples to set
- cData – number of samples to set in rgbData

This function is used to set the custom data; as double precision floating point samples (rgdData) normalized to +/-1 values.
With the custom function option, the data samples (cdData) will be interpolated to the AnalogOut device buffer size. The output value will be Offset + Sample*Amplitude, for instance:
- 0 value sample will output: offset
- +1 value sample will output: offset + amplitude
- -1 value sample will output: offset - amplitude

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

8. AnalogIO

The AnalogIO functions are used to control the power supplies, reference voltage supplies, voltmeters, ammeters, thermometers, and any other sensors on the device. These are organized into channels which contain a number of nodes. For instance, a power supply channel might have three nodes: an enable setting, a voltage level setting/reading, and current limitation setting/reading.

```
FDwfAnalogIOReset(HDWF hdwf);
```

Parameters:
- hdwf – open interface handle on a device

This function resets and configures all AnalogIO instrument parameters to default values for the specified channel.

```
FDwfAnalogIOConfigure(HDWF hdwf);
```

Parameters:
- hdwf – open interface handle on a device

This function is used to configure the instrument and start or stop the AnalogIO instrument.

```
FDwfAnalogIOStatus(HDWF hdwf);
```

Parameters:
- hdwf – open interface handle on a device

This function reads the status of the device and stores it internally. The following status functions will return the information that was read from the device when this function was called.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

FDwfAnalogIOEnableInfo(HDWF hdwf, BOOL* pfSet, BOOL* pfStatus);

Parameters:
- hdwf – open interface handle on a device
- pfSet – returns true when the master enable setting is supported.
- pfStatus – return true when the status of the master enable can be read.

This function is used to verify if Master Enable Setting and/or Master Enable Status are supported for the AnalogIO instrument. The Master Enable setting is essentially a software switch that "enables" or "turns on" the AnalogIO channels. If supported, the status of this Master Enable switch (Enabled/Disabled) can be queried by calling **FDwfAnalogIOEnableStatus**.

FDwfAnalogIOEnableSet(HDWF hdwf, BOOL fMasterEnable);

Parameters:
- hdwf – open interface handle on a device
- fMasterEnable – set TRUE to enable the master switch; FALSE to disable the master switch

This function is used to set the master enable switch.

FDwfAnalogIOEnableGet(HDWF hdwf, BOOL* pfMasterEnable);

Parameters:
- hdwf – open interface handle on a device
- pfMasterEnable – pointer to variable to return the enabled configuration

This function returns the current state of the master enable switch. This is not obtained from the device.

FDwfAnalogIOEnableStatus(HDWF hdwf, BOOL* pfMasterEnable);

Parameters:
- hdwf – open interface handle on a device
- pfMasterEnabled – pointer to variable to return the active status

This function returns the master enable status (if the device supports it). This can be a switch on the board or an overcurrent protection circuit state.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

```
FDwfAnalogIOChannelCount(HDWF hdwf, int* pcChannel);
```

Parameters:
- hdwf – open interface handle on a device
- pcChannel – pointer to variable to return the number of channels

This function returns the number of AnalogIO channels available on the device.

```
FDwfAnalogIOChannelName(HDWF hdwf, int idxChannel, char szName[32], char
szLabel[8]);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- szName – pointer to character array to return the user name
- szLabel – pointer to character array to return the label

This function returns the name (long text) and label (short text, printed on the device) for an AnalogIO channel.

```
FDwfAnalogIOChannelInfo(HDWF hdwf, int idxChannel, int* pnNodes);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- pnNodes – pointer to variable to return the number of node

This function returns the number of nodes associated with the specified channel.

```
FDwfAnalogIOChannelNodeName(HDWF hdwf, int idxChannel, char szNodeName[32], char
szUnits[8]);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- szNodeName – pointer to character array to return the node name
- szUnits – pointer to character array to return the value units

This function returns the node name ("Voltage", "Current"…) and units ("V", "A") for an Analog IO node .

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

FDwfAnalogIOChannelNodeInfo(HDWF hdwf, int idxChannel, ANALOGIO* panalogio);

Parameters:
- hdwf – open interface handle on a device
- idxChannel – channel index
- panalogio – pointer to variable to return the node type

This function returns the supported AnalogIO channel modes. They are returned (by reference) as a bit field. This bit field can be parsed using the IsBitSet Macro. Individual bits are defined using the ACQMODE constants in DWH.h. The acquisition mode selects one of the following modes, ANALOGIO:

- analogioEnable: Enable IO node; used to enable a power supply, reference voltage, etc.
- analogioVoltage: Voltage IO node; used to input/output voltage levels
- analogioCurrent: Current IO node; used to input/output current levels
- analogioTemperature: Temperature IO node; used to retrieve read values from a temperature sensor

FDwfAnalogIOChannelNodeSetInfo(HDWF hdwf, int idxChannel, int idxNode, double* pmin, double* pmax, int* pnSteps);

Parameters:
- hdwf – open interface handle on a device
- idxChannel – Analog IO channel index of the device
- pmin – minimum settable value
- pmax – maximum settable value
- pnSteps – number of steps between minimum and maximum values

This functions returns node value limits. Since a Node can represent many things (Power supply, Temperature sensor, etc), the *Minimum*, *Maximum*, an *Steps* parameters also represent different types of values. In broad terms, the (Maximum – Minimum)/Steps = the number of specific Input/Output values.
FDwfAnalogIOChannelNodeInfo returns the type of values to expect and
FDwfAnalogIOChannelNodeName returns the units of these values.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

```
FDwfAnalogIOChannelNodeSet(HDWF hdwf, int idxChannel, int idxNode, double value);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – Analog IO channel index of the device
- value – value to set

This function is used to set the node value for the specified node on the specified channel.

```
FDwfAnalogIOChannelNodeGet(HDWF hdwf, int idxChannel, int idxNode, double*
pvalue);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – Analog IO channel index of the device
- pvalue – pointer to variable to return the configured value

This function returns the currently set value of the node on the specified channel.

```
FDwfAnalogIOChannelNodeStatusInfo(HDWF hdwf, int idxChannel, int idxNode, double*
pmin, double* pmax, int* pnSteps);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – Analog IO channel index of the device
- pmin – minimum reading value
- pmax – maximum reading value
- pnSteps – number of steps between minimum and maximum values

This function returns node the range of reading values available for the specified node on the specified channel.

```
FDwfAnalogIOChannelNodeStatus(HDWF hdwf, int idxChannel, int idxNode, double*
pvalue);
```

Parameters:
- hdwf – open interface handle on a device
- idxChannel – Analog IO channel index of the device
- pvalue – pointer to variable to return the value reading

This function returns the value reading of the node.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

9. DigitalIO

FDwfDigitalIOReset(HDWF hdwf);

Parameters:
- hdwf – open interface handle on a device

This function resets all DigitalIO instrument parameters to default values. It sets the output enables to zero (tri-state), output value to zero, and configures the DigitalIO instrument.

FDwfDigitalIOConfigure(HDWF hdwf);

Parameters:
- hdwf – open interface handle on a device

This function is used to configure the DigitalIO instrument. This doesn't have to be used if AutoConfiguration is enabled.

FDwfDigitalIOStatus(HDWF hdwf);

Parameters:
- hdwf – open interface handle on a device

This function reads the status and input values, of the device DigitalIO to the PC. The status and values are accessed from the **FDwfDigitalIOInputStatus** function.

FDwfDigitalIOOutputEnableInfo(HDWF hdwf, unsigned int* pfsOutputEnableMask);

Parameters:
- hdwf – open interface handle on a device
- pfsOutputEnableMask – variable to return the OE mask bit field.

This function returns the output enable mask (bit set) that can be used on this device. These are the pins that can be used as outputs on the device.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

```
FDwfDigitalIOOutputEnableSet(HDWF hdwf, unsigned int fsOutputEnable);
```

Parameters:
- hdwf – open interface handle on a device
- fsOutputEnable – output enable bit set

This function is used to enable specific pins for output. This is done by setting bits in the fsOutEnable bit field (1 for enabled, 0 for disabled).

```
FDwfDigitalIOOutputEnableGet(HDWF hdwf, unsigned int* pfsOutputEnable);
```

Parameters:
- hdwf – open interface handle on a device
- pfsOutputEnable – pointer to variable to returns output enable bit set

This function returns a bit field that specifies which output pins have been enabled.

```
FDwfDigitalIOOutputInfo(HDWF hdwf, unsigned int* pfsOutputMask);
```

Parameters:
- hdwf – open interface handle on a device
- pfsOutputMask – variable to return the output value mask

This function returns the settable output value mask (bit set) that can be used on this device.

```
FDwfDigitalIOOutputSet(HDWF hdwf, unsigned int fsOutput);
```

Parameters:
- hdwf – open interface handle on a device
- fsOutput – output enable bit set

This function is used to set the output logic value on all output pins.

```
FDwfDigitalIOOutputGet(HDWF hdwf, unsigned int* pfsOutput);
```

Parameters:
- hdwf – open interface handle on a device
- pfsOutput – pointer to variable to returns output enable bit set

This function returns the currently set output values across all output pins.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

```
FDwfDigitalIOInputInfo(HDWF hdwf, unsigned int* pfsInputMask);
```

Parameters:

- hdwf – open interface handle on a device
- pfsInputMask – variable to return the input value mask

This function returns the readable input value mask (bit set) that can be used on the device.

```
FDwfDigitalIOInputStatus(HDWF hdwf, unsigned int* pfsInput);
```

Parameters:

- hdwf – open interface handle on a device
- pfsInput – variable to return the input value

This function returns the input states of all IO pins. Before calling this function, call the `FDwfDigitalIOStatus` function to read the Digital IO states from the device.

Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

10. Examples

The following examples demonstrate basic use of each API function set.
They are written in pseudo-code, similar to C.

10.1 Device enumeration

```
int cDevice
int cChannel
double hzFreq
char szDeviceName[32]
char szSN[32]
BOOL fIsInUse
HDWF hdwf

// detect connected all supported devices
FDwfEnum(enumfilterAll, &cDevice)
// list information about each device
For idx in 1 to cDevice
  // we use 0 based indexing
  FDwfEnumDeviceName (idx-1, szDeviceName)
  FDwfEnumSN(idx-1, szSN)
  Print("device: "+idx+" name: "+szDeviceName+" "+szSN)
  // before opening, check if the device isn't already opened by other
     application, like: WaveForms
  FDwfEnumDeviceIsOpened(idx-1, fIsInUse)
  If !fIsInUse
    FDwfDeviceOpen(idx-1, &hdwf)
    FDwfAnalogInChannelCount(hdwf, &cChannel)
    FDwfAnalogInFrequencyInfo(hdwf, null, &hzFreq)
    Print("number of analog input channels: "+cChannel+" maximum freq.:"+hzFreq)
    FDwfDeviceClose(hdwf)
    hdwf = hdwfNone
  End if
End for
// before application exit make sure to close all opened devices by this process
FDwfDeviceCloseAll()
```


Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

10.2 Generate sine wave

```
HDWF hdwf;

// open automatically the first available device
FDwfDeviceOpen(-1, &hdwf)
// enable first channel
FDwfAnalogOutEnableSet(hdwf, 0, true)
// set sine function
FDwfAnalogOutFunctionSet(hdwf, 0, funcSine)
// 10kHz
FDwfAnalogOutFrequencySet(hdwf, 0, 10000.0)
// 1.41V amplitude (1Vrms), 2.82V pk2pk
FDwfAnalogOutAmplitudeSet(hdwf, 0, 1.41)
// 1.41V offset
FDwfAnalogOutOffsetSet(hdwf, 0, 1.41)
// start signal generation
FDwfAnalogOutConfigure(hdwf, 0, true)
// it will run until stopped, reset, parameter changed or device closed
Wait(10)
// on close device is stopped and configuration lost
FDwfDeviceClose(hdwf)
```


Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

10.3 Generate custom waveform

```
HDWF hdwf
double rgdSamples[4096]

// open automatically the first available device
FDwfDeviceOpen(-1, &hdwf)
// enable first channel
FDwfAnalogOutEnableSet(hdwf, 0, true)
// set custom function
FDwfAnalogOutFunctionSet(hdwf, 0, funcCustom)
// set custom waveform samples
// normalized to ±1 values
FDwfAnalogOutDataSet(hdwf, 0, rgdSamples, 4096)
// 10kHz waveform frequency
FDwfAnalogOutFrequencySet(hdwf, 0, 10000.0)
// 2V amplitude, 4V pk2pk, for sample value -1 will output -2V, for 1 +2V
FDwfAnalogOutAmplitudeSet(hdwf, 0, 2)
// by default the offset is 0V
// start signal generation
FDwfAnalogOutConfigure(hdwf, 0, true)
// it will run until stopped or device closed
Wait(10)
// on close device is stopped and configuration lost
FDwfDeviceCloseAll()
```


Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

10.4 Single sample read

```
HDWF hdwf
double voltage

// open automatically the first available device
FDwfDeviceOpen(-1, &hdwf)
// enable first channel
FDwfAnalogInChannelEnableSet(hdwf, 0, true)
// set 0V offset
FDwfAnalogInChannelOffsetSet(hdwf, 0, 0)
// set 5V pk2pk input range, -2.5V to 2.5V
FDwfAnalogInChannelRangeSet(hdwf, 0, 5)
// start signal generation
FDwfAnalogInConfigure(hdwf, 0, false)
// wait at least 2 seconds with Analog Discovery for the offset to stabilize,
before the first reading after device open or offset/range change
Wait(2)
// fetch analog input information from the device
FDwfAnalogInStatus(hdwf, false, NULL)
// read voltage input of first channel
FDwfAnalogInStatusSample(hdwf, 0, &voltage)
// close all opened devices by this process
FDwfDeviceCloseAll()
```


Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

10.5 Acquisition

```
HDWF hdwf
STS sts
double rgdSamples[4000]

// open automatically the first available device
FDwfDeviceOpen(-1, &hdwf)
// 20MHz sample rate
FDwfAnalogOutFrequencySet(hdwf, 0, 20000000.0)
// acquire 4k samples
FDwfAnalogInBufferSizeSet(hdwf, 0, 4000)
// enable first channel
FDwfAnalogInChannelEnableSet(hdwf, 0, true)
// set 5V pk2pk input range
FDwfAnalogInChannelRangeSet(hdwf, 0, 5)
// wait at least 2 seconds with Analog Discovery for the offset to stabilize,
before the first reading after device open or offset/range change
Wait(2)
// start
FDwfAnalogInConfigure(hdwf, 0, true)
// wait for the acquisition to be finished
Do{
  FDwfAnalogInStatus(hdwf, true, &sts)
}while(sts != stsDone)
// read the acquired samples of channel 1
FDwfAnalogInStatusData(hdwf, 0 rgdSamples, 4000)
// close the device
FDwfDeviceClose(hdwf)
```


Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

10.6 Analog Discovery Power Supplies

```
HDWF hdwf
STS sts
BOOL fEnabled
double supplyVoltage
double supplyCurrent
double supplyPower
int supplyLoadPercentage

// open automatically the first available device
FDwfDeviceOpen(-1, &hdwf)
// enable positive supply
FDwfAnalogIOChannelNodeSet(hdwf, 0, 0, 1)
// enable negative supply
FDwfAnalogIOChannelNodeSet(hdwf, 1, 0, 1)
// master enable
FDwfAnalogIOEnableSet(hdwf, true)

For 1 to 60
  // wait before the first reading after device open and between reading
  Wait(1)
  // fetch analogIO status from device
  FDwfAnalogIOStatus(hdwf)

  // supply monitor
  FDwfAnalogIOChannelNodeStatusInfo(hdwf, 3, 0, &supplyVoltage)
  FDwfAnalogIOChannelNodeStatusInfo(hdwf, 3, 1, &supplyCurrent)
  supplyPower = supplyVoltage* supplyCurrent
  Print("Total supply power: "+supplyPower+"W")

  supplyLoadPercentage = 100*supplyCurrent/0.2
  Print("Load: "+supplyLoadPercentage+"%")

  // in case of overcurrent condition the supplies are disabled
  FDwfAnalogIOEnableStatus(hdwf, &fEnabled)
  If !fEnabled
    // re-enable supplies
    FDwfAnalogIOEnableSet(hdwf, false)
    FDwfAnalogIOEnableSet(hdwf, true)
  End if
End for
// close the device
FDwfDeviceClose(hdwf)
```


Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

10.7 Analog Discovery System Monitor

```
HDWF hdwf
STS sts
double deviceVoltage
double deviceCurrent
double deviceTemperature

// open automatically the first available device
FDwfDeviceOpen(-1, &hdwf)

For 1 to 60
  // wait before the first reading after device open and between readings
  // the update rate is approximately 1Hz
  Wait(1)
  // fetch analog IO status from device
  FDwfAnalogIOStatus(hdwf)
  // get system monitor readings
  FDwfAnalogIOChannelNodeStatusInfo(hdwf, 2, 0, &deviceVoltage)
  FDwfAnalogIOChannelNodeStatusInfo(hdwf, 2, 1, &deviceCurrent)
  FDwfAnalogIOChannelNodeStatusInfo(hdwf, 2, 2, &deviceTemperature)
  Print("Device USB supply voltage: "+deviceVoltage+"V")
  Print("Device USB supply current: "+deviceVoltage+"A")
  Print("Device temperature: "+deviceVoltage+"C")
End for
// close the device
FDwfDeviceClose(hdwf)
```


Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.

10.8 Digital IO

```
HDWF hdwf

// open automatically the first available device
FDwfDeviceOpen(-1, &hdwf)
// enable output/mask on 8 LSB IO pins, from DIO 0 to 7
FDwfDigitalIOOutputEnableSet(hdwf, 0x00FF)
// set value on enabled IO pins
FDwfDigitalIOOutputSet(hdwf, 0x12)
// fetch digital IO information from the device
FDwfDigitalIOStatus (hdwf)
DWORD dwRead
// read state of all pins, regardless of output enable
FDwfDigitalIOInputStatus(hdwf, 0, &dwRead)
// close all opened devices by this process
FDwfDeviceCloseAll()
```


Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners.